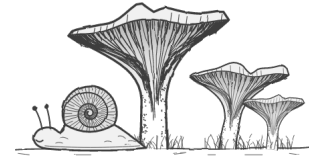


Structured Masked Attention (SMA)

Mikkel Paltorp (mpasc@dtu.dk), 28 July 2025

Department of Applied Mathematics and Computer Science, Technical University of Denmark



Masked Attention (MA) is given by the relation [1]

$$\mathbf{M} = \mathbf{B} \odot \mathbf{L}, \quad (1)$$

where $\mathbf{L}$ is a mask applied to the matrix $\mathbf{B}$ using the elementwise Hadamard product (denoted by $\odot$). When the mask is **structured** one can most often apply multiplication with $\mathbf{M}$ efficiently, and we refer to the masked attention as Structured Masked Attention (SMA). In the simplest case of $\mathbf{L}$ being a lower triangular matrix filled with ones the SMA reduces to

$$\mathbf{M} = \mathbf{B} \odot \mathbf{L} = \text{tril}(\mathbf{B}), \quad (2)$$

which can be viewed as a weighted cumulative sum. While the above is a **structured** computation, it is not efficient unless $\mathbf{B}$ itself has some structure. An example of when $\mathbf{B}$ is structured is when it is of low-rank (i.e., $\mathbf{B} = \mathbf{U}\mathbf{V}^\top$). The resulting SMA is a semiseparable matrix for which multiplication can be applied in linear time [2].

In general if $\mathbf{L}$ is a low-rank matrix (of rank p) then multiplication with $\mathbf{M}$ scales as p -times the scaling of multiplication with $\mathbf{B}$ as

$$\mathbf{M}\mathbf{x} = (\mathbf{B} \odot \mathbf{U}\mathbf{V}^\top)\mathbf{x} = \sum_{i=1}^p \text{diag}(\mathbf{u}_i)\mathbf{B} \text{diag}(\mathbf{v}_i)\mathbf{x}, \quad (3)$$

for which we see that we need to perform p multiplications with $\mathbf{B}$ as well as $2p$ diagonal multiplications.

julia

```
1 using LinearAlgebra
2 using Random
3 using Test
4 Random.seed!(42)
5 n, p = 10, 2
6 U, V = randn(n,p), randn(n,p)
7 B = rand(n,n)
8 M = B .* (U*V')
9 x = randn(n)
10 @test M*x ≈ sum(i -> Diagonal(U[:,i])*(B*(Diagonal(V[:,i])) * x), 1:p)
```

1. State-space models as structured matrices

First we recap that a state-space model is given by the equations

$$\begin{aligned} \mathbf{h}_i &= \mathbf{A}_i \mathbf{h}_{i-1} + \mathbf{B}_i \mathbf{x}_i, \\ \mathbf{y}_i &= \mathbf{C}_i^\top \mathbf{h}_i, \end{aligned} \quad (4)$$

An alternative way to view the state-space model is through the lens of structured matrices. We take $\mathbf{h}_0 = \mathbf{B}_0 \mathbf{x}_0$, corresponding to a zero state before the first input. In particular a state-space model up until time step T can also be written using block matrices as well as block bidiagonal matrices as

Contents

1. State-space models as structured matrices
2. State-space models as SMAs

Bibliography

$$\underbrace{\begin{bmatrix} \mathbf{I} & & & \\ -\mathbf{A}_1 & \mathbf{I} & & \\ & \ddots & \ddots & \\ & & -\mathbf{A}_{T-1} & \mathbf{I} \end{bmatrix}}_{\mathbf{A}} \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{T-1} \end{bmatrix} = \begin{bmatrix} \mathbf{B}_0 \mathbf{x}_0 \\ \mathbf{B}_1 \mathbf{x}_1 \\ \vdots \\ \mathbf{B}_{T-1} \mathbf{x}_{T-1} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{B}_0 & & & \\ & \mathbf{B}_1 & & \\ & & \ddots & \\ & & & \mathbf{B}_{T-1} \end{bmatrix}}_{\mathbf{B}} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{T-1} \end{bmatrix} \quad (5)$$

We know that by simply iterating forwards in time we can compute the states in linear time. For the following generator representation, assume that each transition matrix $\mathbf{A}_i$, $i = 1, \dots, T-1$, is invertible. A structured representation of the inverse of the bidiagonal matrix $\mathbf{A}$ can be constructed and applied in linear time for fixed state dimension, using

$$\mathbf{A}^{-1} = \text{tril} \left(\underbrace{\begin{bmatrix} \mathbf{I} \\ \mathbf{A}_1 \\ \mathbf{A}_2 \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_{T-1} \dots \mathbf{A}_1 \end{bmatrix}}_{\mathbf{U}} \underbrace{\begin{bmatrix} \mathbf{I} \\ \mathbf{A}_1^{-\top} \\ (\mathbf{A}_2 \mathbf{A}_1)^{-\top} \\ \vdots \\ (\mathbf{A}_{T-1} \dots \mathbf{A}_1)^{-\top} \end{bmatrix}}_{\mathbf{V}^\top} \right)^\top, \quad (6)$$

This type of matrix structure is called semiseparable [2]. Using this structured representation of $\mathbf{A}^{-1}$ we can compute the hidden states efficiently as

$$\begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{T-1} \end{bmatrix} = \text{tril} \left(\underbrace{\begin{bmatrix} \mathbf{I} \\ \mathbf{A}_1 \\ \mathbf{A}_2 \mathbf{A}_1 \\ \vdots \\ \mathbf{A}_{T-1} \dots \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} \mathbf{I} \\ \mathbf{A}_1^{-\top} \\ (\mathbf{A}_2 \mathbf{A}_1)^{-\top} \\ \vdots \\ (\mathbf{A}_{T-1} \dots \mathbf{A}_1)^{-\top} \end{bmatrix}^\top}_{\mathbf{A}^{-1}} \right)^\top \begin{bmatrix} \mathbf{B}_0 & & & \\ & \mathbf{B}_1 & & \\ & & \ddots & \\ & & & \mathbf{B}_{T-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{T-1} \end{bmatrix}, \quad (7)$$

Similarly, the output $\mathbf{y}$ can be computed by applying the output matrices $\mathbf{C}_i$ in a blocked fashion to the hidden states, i.e.

$$\begin{bmatrix} \mathbf{y}_0 \\ \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_{T-1} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{C}_0^\top & & & \\ & \mathbf{C}_1^\top & & \\ & & \ddots & \\ & & & \mathbf{C}_{T-1}^\top \end{bmatrix} \begin{bmatrix} \mathbf{I} & & & \\ -\mathbf{A}_1 & \mathbf{I} & & \\ & \ddots & \ddots & \\ & & -\mathbf{A}_{T-1} & \mathbf{I} \end{bmatrix}^{-1}}_{\mathbf{M}} \begin{bmatrix} \mathbf{B}_0 & & & \\ & \mathbf{B}_1 & & \\ & & \ddots & \\ & & & \mathbf{B}_{T-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_{T-1} \end{bmatrix} \quad (8)$$

julia

```
1 using SymSemiseparableMatrices
2 using BlockBandedMatrices
3 using LinearAlgebra
4 using Random
5 using Test
6 Random.seed!(42)
7
8 T = 10      # Sequence length
9 n = 6       # State size
10 input_dim = 1 # Dimension of forcing term
11
12 A_blks = [rand()*I(n) for _ in 1:T-1] # Diagonal dynamics in Mamba 2
13 D_blks = [I(n)/1.0 for i in 1:T]      # Diagonal blocks are identity
14 C_blks = [rand(n,1) for _ in 1:T]     # Measurements are scalars
15 B_blks = [rand(n,input_dim) for _ in 1:T] # Inputs are scalars
16 # Defining zeros blocks for the matrices
17 A_zero_blks = [zeros(n,n) for b in A_blks]
18 C_zero_blks = [zeros(n,1) for _ in 1:T-1]
19 B_zero_blks = [zeros(n,input_dim) for _ in 1:T-1]
20 # Defining the block matrices
```

```

21 A = BlockTridiagonal(-A_blks,D_blks,A_zero_blks)
22 C = BlockTridiagonal(C_zero_blks,C_blks,C_zero_blks)
23 B = BlockTridiagonal(B_zero_blks,B_blks,B_zero_blks)
24
25 # Computing states by iterating forward
26 x_blks = [randn(input_dim) for _ in 0:T-1] # input data
27 h_blks = [B_blks[1]*x_blks[1]] # initial hidden state
28 for i in 2:T
29     push!(h_blks, A_blks[i-1]*h_blks[i-1] + B_blks[i]*x_blks[i])
30 end
31 y_blks = [C'*h for (C,h) in zip(C_blks,h_blks)]
32
33 # Computing states using semiseparable matrices
34 Ai_blks = [prod(reverse(A_blks[1:i-1]),init=1.0*I(n)) for i in 1:T]
35 U = vcat(Ai_blks...)
36 V = vcat(inv.(transpose.(Ai_blks))...)
37 # "SymSemiseparableCholesky" represents the matrix A^{-1} = tril(UV')
38 Ai = SymSemiseparableCholesky(U', V')
39 x = vcat(x_blks...) # Collecting input data
40
41 @test Ai*(B*x) ≈ vcat(h_blks...) # Checking hidden states
42 @test C'*(Ai*(B*x)) ≈ vcat(y_blks...) # Checking measurement

```

2. State-space models as SMAs

In this section we aim to derive that the simplified SSM in the Mamba-2 paper is a special case of Structured Masked Attention (SMA) [1], i.e., that the multiplication

$$\mathbf{M}\mathbf{x} = \begin{bmatrix} \mathbf{C}_0 & & & \\ & \mathbf{C}_1 & & \\ & & \ddots & \\ & & & \mathbf{C}_{T-1} \end{bmatrix}^\top \left(\mathbf{A}^{-1} \left(\begin{bmatrix} \mathbf{B}_0 & & & \\ & \mathbf{B}_1 & & \\ & & \ddots & \\ & & & \mathbf{B}_{T-1} \end{bmatrix} \mathbf{x} \right) \right), \quad (9)$$

can be written differently as

$$\mathbf{M}\mathbf{x} = (\mathbf{B} \odot \mathbf{L})\mathbf{x}, \quad (10)$$

The SSM in the Mamba-2 paper restricts the dynamics of $\mathbf{A}$ to be scalar-times-identity dynamics in order for the mask to be structured [1]. In short this restriction means that the dynamics for all hidden states are independent but equal.

$$\begin{aligned} \mathbf{h}_t &= (a_t \mathbf{I})\mathbf{h}_{t-1} + \mathbf{b}_t x_t, \\ \mathbf{y}_t &= \mathbf{c}_t^\top \mathbf{h}_t. \end{aligned} \quad (11)$$

From a practical point-of-view this means that we can collect each index of the dynamics and treat them separately. The resulting $\mathbf{A}$ -matrix can be described by a Kronecker product (depending on how we organize the states it's either $\mathbf{A} = \mathbf{a} \otimes \mathbf{I}_n$ or $\mathbf{A} = \mathbf{I}_n \otimes \mathbf{a}$). In the following we choose to separate the states, resulting in $\mathbf{A}$ having the form

$$\mathbf{A} = \mathbf{I}_n \otimes \mathbf{a}, \quad \mathbf{a} = \begin{bmatrix} 1 & & & \\ -a_1 & 1 & & \\ & \ddots & \ddots & \\ & & -a_{T-1} & 1 \end{bmatrix}, \quad (12)$$

For the following scalar generator representation, assume that $a_i \neq 0$ for $i = 1, \dots, T-1$. Using that the inverse of a Kronecker product is the Kronecker product of the inverses it follows that

$$\mathbf{A}^{-1} = \mathbf{I}_n \otimes \mathbf{a}^{-1}, \quad \mathbf{a}^{-1} = \text{tril} \left(\underbrace{\begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_{T-1} \dots a_1 \end{bmatrix}}_{\mathbf{u}} \underbrace{\begin{bmatrix} 1 \\ a_1^{-1} \\ \vdots \\ (a_{T-1} \dots a_1)^{-1} \end{bmatrix}}_{\mathbf{v}^\top} \right)^\top, \quad (13)$$

where we further used that the inverse of a bidiagonal matrix is semiseparable. Furthermore, we have to re-arrange $\mathbf{B}$ and $\mathbf{C}$ which results in

$$\mathbf{B} = \begin{bmatrix} \text{diag}(\mathbf{b}^{(1)}) \\ \vdots \\ \text{diag}(\mathbf{b}^{(n)}) \end{bmatrix}, \quad \mathbf{b}^{(k)} = \begin{bmatrix} b_0^{(k)} \\ \vdots \\ b_{T-1}^{(k)} \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \text{diag}(\mathbf{c}^{(1)}) \\ \vdots \\ \text{diag}(\mathbf{c}^{(n)}) \end{bmatrix}, \quad \mathbf{c}^{(k)} = \begin{bmatrix} c_0^{(k)} \\ \vdots \\ c_{T-1}^{(k)} \end{bmatrix}, \quad (14)$$

where $b_t^{(k)}$ and $c_t^{(k)}$ denote the k th entries of $\mathbf{b}_t$ and $\mathbf{c}_t$. The final multiplication will therefore look as

$$\underbrace{\mathbf{C}^\top \mathbf{A}^{-1} \mathbf{B}}_{\mathbf{M}} \mathbf{x} = \left(\sum_{k=1}^n \text{diag}(\mathbf{c}^{(k)}) \mathbf{a}^{-1} \text{diag}(\mathbf{b}^{(k)}) \right) \mathbf{x} = \left(\sum_{k=1}^n (\mathbf{c}^{(k)} \mathbf{b}^{(k)\top}) \odot \mathbf{a}^{-1} \right) \mathbf{x}. \quad (15)$$

Finally, using the properties of the Hadamard product we get to the Structured Masked Attention form that we were looking for

$$\begin{aligned} (\mathbf{C}^\top \mathbf{A}^{-1} \mathbf{B}) \mathbf{x} &= \left(\sum_{k=1}^n (\mathbf{c}^{(k)} \mathbf{b}^{(k)\top}) \odot \mathbf{a}^{-1} \right) \mathbf{x} \\ &= \left(\left(\begin{bmatrix} \mathbf{c}^{(1)} & \dots & \mathbf{c}^{(n)} \end{bmatrix} \begin{bmatrix} \mathbf{b}^{(1)\top} \\ \vdots \\ \mathbf{b}^{(n)\top} \end{bmatrix} \right) \odot \mathbf{a}^{-1} \right) \mathbf{x}. \end{aligned} \quad (16)$$

This means that the SSM dynamics can be interpreted as a structured masked attention mechanism. Note that in the case of the dynamics being independent but different (i.e., we would have $\mathbf{a}^{(k)}$ rather than just $\mathbf{a}$) the SSM dynamics would result in a sum of masked attentions, i.e.,

$$(\mathbf{C}^\top \mathbf{A}^{-1} \mathbf{B}) \mathbf{x} = \left(\sum_{k=1}^n (\mathbf{c}^{(k)} \mathbf{b}^{(k)\top}) \odot (\mathbf{a}^{(k)})^{-1} \right) \mathbf{x} \quad (17)$$

julia

```
1 using SymSemiseparableMatrices
2 using LinearAlgebra
3 using Random
4 using Test
5 Random.seed!(42)
6
7 T = 10           # Sequence length
8 n = 6           # State size
9
10 # Here we treat the matrices in terms of their states and not sequence lengths
11 a_blks = rand(T-1)
12 a = Bidiagonal(ones(T), -a_blks, :L)
13 A = kron(I(n), a)      # Kronecker product with identity
14
15 # The blocks are now size equal to the sequence and a block for each state!
16 B_blks = [rand(T) for _ in 1:n]
17 C_blks = [rand(T) for _ in 1:n]
18 # Collecting the blocks into the B and C
19 B = vcat(Diagonal.(B_blks)...)
20 C = vcat(Diagonal.(C_blks)...)
21
22 # We want to see if the full matrix M = C'*(A\B) can be written as
```

```

23 # structured masked attention, i.e.,
24 # M = (CB')\circ a^{-1}. For this we start by computing a^{-1}
25 ai = inv(a) # We ignore here that inv(a) is semiseparable
26 @test C'*(A\B) ≈ sum(i-> Diagonal(C_blks[i])*ai*Diagonal(B_blks[i]), 1:n)
27 @test C'*(A\B) ≈ sum(i->(C_blks[i]*B_blks[i]) .* ai, 1:n)
28
29 # We can collect the terms and write it as Structured Masked Attention!
30 Cn = hcat(C_blks...)
31 Bn = hcat(B_blks...)
32 @test C'*(A\B) ≈ (Cn*Bn') .* ai
33
34 # We can apply the semiseparable structure of the inverse when multiplying!
35 ai_blks = [prod(a_blks[1:i-1],init=1.0) for i in 1:T]
36 u = vcat(ai_blks...)
37 v = vcat(inv.(ai_blks)...)
38 # "SymSemiseparableCholesky" represents the matrix A^{-1} = tril(UV')
39 ais = SymSemiseparableCholesky(u',v')
40 # Efficient products using the structure of "ais" and diagonal B and C
41 x = randn(T)
42 @test C'*(A\B*x) ≈ sum(i-> C_blks[i].*(ais*(B_blks[i].*x)), 1:n)

```

Bibliography

- [1] T. Dao and A. Gu, “Transformers are SSMs: Generalized Models and Efficient Algorithms Through Structured State Space Duality,” in *International Conference on Machine Learning (ICML)*, 2024. [Online]. Available: <https://arxiv.org/abs/2405.21060>
- [2] M. S. Andersen and T. Chen, “Smoothing Splines and Rank Structured Matrices: Revisiting the Spline Kernel,” *SIAM Journal on Matrix Analysis and Applications*, vol. 41, no. 2, pp. 389–412, 2020, doi: 10.1137/19M1267349.